基于分段的移动对象轨迹简化算法

张　甜，杨智应

(上海海事大学 信息工程学院, 上海 201306)

摘　要：GPS 的高采样率使轨迹的数据规模巨大，在实际应用中难以处理，需要依赖轨迹简化算法对原始数据进行压缩。针对此问题，提出了一种新的基于速度分段的轨迹简化算法，即 STS 算法，在保留速度特征的同时保留了给定轨迹的时空特征。STS 算法将速度值分组成若干间隔，将轨迹分割成速度保留段，计算各轨迹段的 SED 阈值，通过在每个子轨迹段上应用 TD-TR 算法导出简化的轨迹。通过真实的数据集进行广泛实验，验证所提出的算法比 ATS 算法具有更好的性能。

关键词：移动对象；轮廓系数；TD-TR 算法；CART 决策树；轨迹简化

中图分类号：TP301.6　　doi: 10.3969/j.issn.1001-3695.2018.02.0090

Segmentation-based trajectory simplification algorithm

Zhang Tian[1], Yang Zhiying[2]

(*School of Information Engineering*, *Shanghai Maritime University*, *Shanghai* 201306, *China*)

Abstract: The high sampling rate of GPS makes the data set of the trajectory huge, which is difficult to handle in practical applications. It needs to rely on trajectory simplification algorithm to compress the original data. Aiming at this problem, a new simplification algorithm based on speed segmentation was proposed, namely STS algorithm, which preserved the spatiotemporal characteristics while preserving the velocity characteristics of a given trajectory. The STS algorithm divided the velocity values into several intervals and divides the trajectory into velocity-preserving segments. It calculated the SED threshold of each trajectory segment and derives a simplified trajectory by applying the TD-TR algorithm on each sub-trajectory segment. Extensive experiments with real datasets demonstrate that the proposed algorithm has better performance than ATS algorithms.

Key words: moving object; silhouette coefficient; TD-TR algorithm; CART; trajectory simplification

0　引言

近年来，全球范围内销售的 GPS 设备数量大幅增长。研究市场格局的信息技术公司 Canalys 在报告中称，2006 年至 2007 年期间 GPS 装置的销售数量增长了 116%[1]。以 GPS 为基础的移动设备所构建的基于位置的服务和应用代表着迅速扩大的消费者市场。2009 年，预估有 2700 万台配备 GPS 的智能手机在全球各地销售，仅在消费市场上就将全球 GPS 用户群带到至少 6800 万人[2]。这些设备有能力产生、存储和传输轨迹数据。轨迹被定义为由位置（纬度，经度）和时间信息组成的三元组数据流。

目前基于位置的应用程序在使用轨迹数据时主要存在三个问题。首先，大量的轨迹数据的存储可能会快速地淹没可用的数据存储空间。例如，如果轨迹点数据以每 10 s 的时间间隔被收集，那么 1GB 的存储容量仅能存储约 4 000 个轨迹点[3]。因此，如何有效地存储和查询 GPS 轨迹是一个值得研究的领域[3-6]。其次，通过蜂窝或卫星网络发送大量轨迹数据的成本非常高，一般每兆字节在 5~7 美元[7]。因此，假设追踪 4,000 辆车的轨迹，一天的成本约为 5 000~7 000 美元，每年约为 180 000~2,500 000 美元。最后，随着轨迹数据的规模增大，从数据中检测有效的特征信息将变得更加困难。缩减轨迹数据的规模有可能加速轨迹模式的挖掘[8]。

1　相关工作

1.1　轨迹简化算法的研究现状

最早的基于距离的轨迹压缩算法是 Douglas 和 Peucker 提出的 Douglas-Peucker 算法（简称 DP 算法）[9]，其递归地选择垂直距离大于给定阈值的点，直到所有保留点满足条件。后来 Meratnia 和 Rolf 提出了 TD-TR 算法[3]，是一种自顶向下的时间比算法，它通过用 SED 代替垂直距离来充分考虑时空特征。TD-TR 算法的基本思想是：给定轨迹 T 和参数 ε，通过重复添加 T 中的点得到一个新的轨迹 T'，直到 T' 的时间-同步欧氏

收稿日期：2018-02-23；**修回日期：**2018-04-04

作者简介：张甜（1994-），女，江苏徐州人，硕士研究生，主要研究方向为移动对象轨迹简化（zhangtiantian_0206@163.com）；杨智应（1964-），男，教授，博士，主要研究方向为算法与复杂性、移动计算、分布式计算与软件工程.

距离（SED）小于 ε 。该算法最初使用轨迹的第一个和最后一个点来近似原始轨迹。然后，重复选择引起最大 SED 误差的点，并使用该点来更准确地近似原始轨迹的过程。当最大 SED 误差小于 ε 时，该过程停止。

Wu 等人提出了 Opening Window Time-Ratio 算法[10]，使用 SED 代替垂直距离来考虑时间特征。2013 年，Long 等人[11]提出了基于方向的轨迹简化算法，通过给定的角度阈值来简化轨迹，使求出的简化轨迹最大角度值不大于给定的角度阈值。在文献[11]的基础上，文献[12]提出了简化轨迹的数量在不大于给定值的情况下，使得简化轨迹与原始轨迹方向阈值最小的简化算法。Deng 等人[13]基于文[11]提出了基于方向的实时简化算法。称这类轨迹简化算法为基于方向的轨迹简化算法(DPTS)。

在最近的研究中，Lin 等人[14]考虑轨迹的速度特征，提出了速度保留的 ATS 算法。ATS 算法首先使用肘方法确定 k 值将速度值聚类分段表示，将原始轨迹分解成速度保留子轨迹。然后根据基于最小长度原理（MDL）的阈值决策模型，从而得出不同子轨迹的空间误差阈值。最后使用 Douglas-Peuker 算法用于导出简化轨迹。ATS 算法在将速度值聚类过程中使用 Elbow 方法需要人为判断，无法直接得到 K-means 聚类所需 k 值，对分段后的轨迹使用 Douglas-Peucker 算法推导出简化轨迹。本文对其进行改进，使用轮廓系数法计算 k 值实现批量化聚类，并在简化过程中加入时间维度，使用 TD-TR 算法推导出简化轨迹，进一步考虑了轨迹的时空特性，减小 SED 误差。

1.2 定义与记号

定义 1 轨迹点。定义 p 为轨迹点，表示为一个三元组，即 $p=(x,y,t)$，组内对应属性分别代表该点的经度、纬度、时间。

定义 2 原始轨迹。原始轨迹 T 为若干有序轨迹点的集合。$T=(p_1,p_2,\ldots,p_n)(i\in[1,n])$，其中 $p_i=(x_i,y_i,t_i)$ 且 $t_1<t_2<\ldots<t_i<t_n$。

原始轨迹 T 中轨迹点的个数用 $|T|$ 表示，则 T 中轨迹段的个数为 $|T|-1$ 或 $n-1$。

定义 3 简化轨迹。原始轨迹 T 经过算法删除若干轨迹点（除首尾点），简化轨迹 T_{sim} 是简化后有序轨迹点的集合。定义为 $T_{sim}=(p_{s_1},p_{s_2},p_{s_3},\ldots,p_{s_m})$ $(1\le s_1<s_2<s_m\le n 且 m\le n)$。

简化轨迹 T_{sim} 中轨迹点的个数用 $|T_{sim}|$ 表示，则 T_{sim} 中轨迹段的个数为 $|T_{sim}|-1$ 或 $m-1$。

定义 4 轨迹段。原始轨迹 T 或简化轨迹 T_{sim} 中，对轨迹点 p_i 和 p_j $(1\le i\le j\le n)$ 的连线，用 $\overline{p_ip_j}$ 表示。如果 $j-i=1$，则认为 p_i 和 p_j 是相邻的，相邻两点之间所连的线段称为轨迹 T 或轨迹 T_{sim} 中的轨迹段。

定义 5 轨迹段的长度、平均速度。原始轨迹 T 或简化轨迹 T_{sim} 中，对轨迹点 p_i 和 p_j $(1\le i<j\le n)$，若 p_i 和 p_j 相邻，则轨迹段的长度用 $d(\overline{p_ip_j})$ 表示，相应轨迹段的平均速度用 $v(\overline{p_ip_j})$ 表示。

$$v(\overline{p_ip_j})=\frac{d(\overline{p_ip_j})}{p_j.t_j-p_i.t_i}$$

1.3 基于分段的轨迹简化问题定义

速度保留轨迹简化问题定义如下。给定原始轨迹 T 和如表 1 所示的速度-误差表，其包含速度间隔和 SED 误差阈值的集合，推导出简化轨迹 $T_{sim}=(p_{s_1},p_{s_2},p_{s_3},\ldots,p_{s_m})$，其满足以下要求：a）对于 T_{sim} 中的轨迹段，原始轨迹点与其相对应的简化后轨迹之间的时间-同步欧式距离小于 ε，其中 ε 为 $v(\overline{p_ip_j})$ 所在速度间隔中相应的 SED 误差阈值。b）$|T_{sim}|$ 最小化。

表 1 速度-误差表

速度值	误差阈值
v_1-v_2	ε_1
v_2-v_3	ε_2
v_3-v_4	ε_3

速度—误差表是该问题的输入之一，它表示在不同的速度下应采用不同的SED误差阈值。简化的轨迹可以理解为通过使用尽可能少的点来保持位置和速度特征。

2 基于分段的轨迹简化算法

2.1 STS 算法的框架描述

在本节中，提出了基于分段的轨迹简化算法（STS）。简而言之，STS 算法首先从原始轨迹的观察中导出速度误差表，然后使用该表导出简化的轨迹。STS 算法的框架如图 1 所示。首先，从速度值的分布中找出代表性的速度间隔。然后，基于这些代表性的速度间隔，通过它们将整个轨迹划分成若干子轨迹，每个子轨迹具有与其他子轨迹具有显著不同的速度值。根据速度-误差表和子轨迹的平均速度来确定合适的阈值。最后，通过 TD-TR 算法推导出简化的子轨迹，通过连接简化后子轨迹从而获得简化轨迹。STS 算法的细节将在以下部分进行描述。

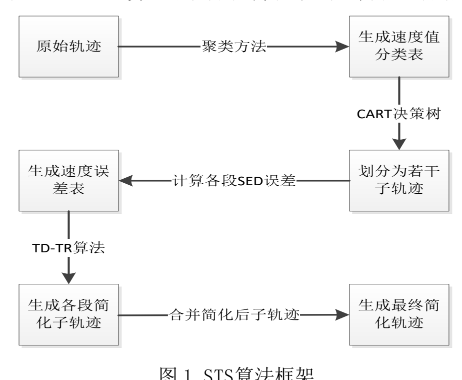


图 1 STS算法框架

2.2 导出代表速度区间

本节介绍了构造速度-误差表时，设置速度间隔的方法。为了确定这些速度间隔的分割方法，本文将聚类方法应用于轨迹数据集的速度特征，从而将速度值划分为若干组。使用 K-means 对速度值进行聚类以获得具有代表性的速度间隔，其中 k 值是数据所分类数，应由用户给出，但从数据集属性的先验知识无法判断 k 的合适值。因此，本文使用了轮廓系数方法[15]来确定 k 的取值。轮廓系数旨在将某个对象与自己的簇的相似程度和与其他簇的相似程度进行比较。轮廓系数最高的簇的数量表示簇的数量的最佳选择。轮廓系数结合了凝聚度和分离度，其计算步骤如下：

a)对于第 i 个对象，计算它到所属簇中所有其他对象的平均距离，即 i 向量到同一簇内其他点不相似程度的平均值，记 $average(i)$（体现凝聚度）

b)对于第 i 个对象和不包含该对象的任意簇，计算该对象到给定簇中所有对象的平均距离，即 i 向量到其他簇的平均不相似程度的最小值，记为 $\min(i)$（体现分离度）。

c)第 i 个对象的轮廓系数计算公式为

$$S(i)=\frac{\min(i)-average(i)}{\max\{average(i),\min(i)\}}$$

由此可得，轮廓系数取值为[-1, 1]，其值越大表示聚类效果越好，当计算结果接近 0 时，表明样本 i 在两个簇的边界上。当计算结果为负时，表明 $average(i)>\min(i)$，样本 i 被分配到错误的簇中，更应该被分配到其他簇中。

如图 2 所示，示例中为了求得 13 个点的最佳聚类 k 值，将 k 值取为 2、3、4、5、8 并分别计算其轮廓系数，当 k=3 时计算出轮廓系数为 0.722，此时轮廓系数最大，根据轮廓系数法原理，其值越大表示聚类效果越好，所以示例中将 k 值取 3 为最佳聚类 k 值。

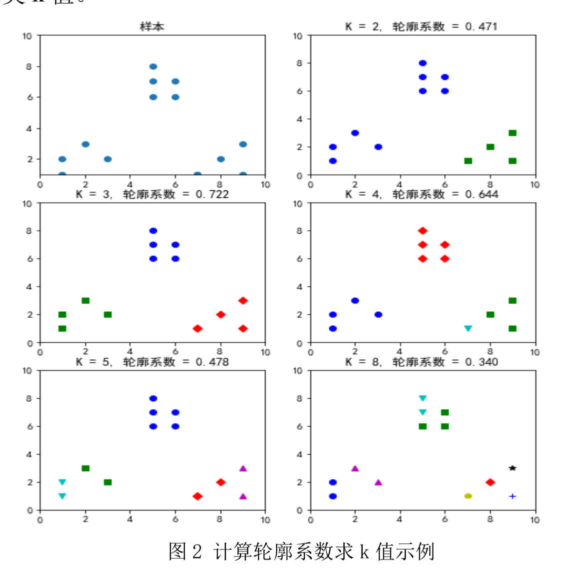


图 2 计算轮廓系数求 k 值示例

2.3 分割原始轨迹

本节描述了如何将原始轨迹划分成若干子轨迹的方法，使得子轨迹之间的速度值具有显着差异。为了达到这个目的，使用 CART 决策树方法[16]判断轨迹是否应分为两部分及如何分割。为了在本文问题中利用 CART 决策树方法的 Gini 系数分类法，原始轨迹 $T=(p_1,p_2,...,p_n)$ 被转换为 $T^v=(L_1,L_2,...,L_{n-1})$，其中 L_i 为轨迹点速度值在速度分类区间中的所属类，即速度标签。速度 $v(p_i,p_{i+1})$ 均包含在此中。例如，给定 $v(p_2,p_3)=28$ 及表 2 中的速度值分类区间，L_2 的值可以被推导出为 1。Gini 系数定义为

$$Gini(T^v)=1-\sum_{i=1}^{k}\left(\frac{N(L_i)}{k}\right)^2$$

，其中 k 为速度标签 T^v 的总大小，$N(\cdot)$ 为 T^v 中 L_i 相等时出现的数量总和，$\frac{N(L_i)}{k}$ 表示各类速度标签在 T^v 中出现的频率。例如，$T=(p_1,p_2,...,p_{13})$，假设其转换后的结果 T^v 为 $T^v=(L_1,L_2,...,L_{12})=(0,0,1,0,0,0,1,1,2,2,2,2)$。原始轨迹点数为 13，所以转换后速度标签数为 12，$k=12$，$L_1=L_2=L_4=L_5=L_6=0$，所以 $N(0)=5$，$L_3=L_7=L_8=1$，所以 $N(1)=3$，$L_9=L_{10}=L_{11}=L_{12}=2$，所以 $N(2)=4$。数据 L_i 计算所得的 Gini 系数越大，代表 L_i 在整体数据集合中的差异性越大。因此，在本文的设置中，所需的子轨迹是基尼系数小的轨迹。划分原始轨迹的方法在算法 1 中列出。

算法 1 分割原始轨迹（$tr_partition$ 算法）

输入： 原始轨迹 T，分割阈值 $Gini_{\min}$

输出： $S_T=\{T_{1,m},T_{m,o},...,T_{p,n}\}$

1: $S_T=\varnothing$

2: $T^v=\{L_1,L_2,...,L_{n-1}\}$;

3: **if** $Gini(T_v)\le Gini_{\min}$ **then**

return $\{T_v\}$;

end

$i = \arg\min_{0<i<n}\left(Gini_{split}\left(T^v_{1,i,n-1}\right)\right)$;

$S_T = S_T \cup tr_partition\left(T_{0,i}, Gini_{\min}\right) \cup tr_partition\left(T_{i,n}, Gini_{\min}\right)$;

return S_T;

表 2　速度分类

类	速度（km/h）
0	0-25
1	25-47
2	47-88
3	>88

a)使用最小 $Gini_{split}$ 将 T^v 分割成两个轨迹段 $T^v_{1,i} = \left(L_1, L_2, ..., L_i\right)$ 及 $T^v_{i+1,n-1} = \left(L_{i+1}, L_{i+2}, ..., L_{n-1}\right)$，$Gini_{split}$ 的定义如下：

$Gini_{split}\left(T^v_{1,i,n-1}\right) = \frac{N_1}{N} Gini\left(T^v_{1,i}\right) + \frac{N_2}{N} Gini\left(T^v_{i+1,n-1}\right)$ 其中 N，N_1, N_2 是 T^v、$T^v_{1,i}$ 和 $T^v_{i+1,n-1}$ 的大小，且 $N_1 + N_2 = N$。

b)对于 $T^v_{1,i}$ 和 $T^v_{i+1,n-1}$，计算它们的基尼系数 $Gini\left(T^v_{1,i}\right)$ 及 $Gini\left(T^v_{i+1,n-1}\right)$。对于每一个子轨迹，如果其基尼系数大于给定的阈值 $Gini_{\min}$，递归地进行操作 a)。

2.4　计算 SED 误差阈值

本节讨论如何为每个速度间隔选择 SED 误差阈值 ε。在此步骤之后，可以建立速度-误差表。由于这些子轨迹具有相似的速度标签，所以观察这些子轨迹可以帮助每个代表速度间隔决定适当的 SED 误差阈值。SED (Synchronized Euclidean Distance) 即同步欧式距离，如图 3 所示，SED 距离为 P_i 与 P_i' 之间的欧氏距离，其中 P_i' 坐标位置的具体计算公式如下。

$$\Delta e = t_e - t_s$$
$$\Delta i = t_i - t_s$$

$$x_i' = x_s + \frac{\Delta i}{\Delta e}\left(x_e - x_s\right)$$
$$y_i' = y_s + \frac{\Delta i}{\Delta e}\left(y_e - y_s\right)$$

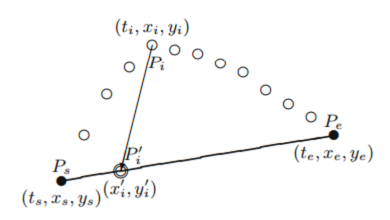


图 3 SED 距离的定义

简化后较低的误差和较小的轨迹规模通常是互相制约的，因此难以同时实现。为了在这两个因素之间取得良好的平衡，定义了一个 Balance 值，表示每一次迭代循环的 SED 误差比分与轨迹规模比分之间的和，Balance 值越低表示更好的拟合简化。因此，我们可以获得不同速度变化范围内子轨迹的优化 ε。在开始迭代时初始的子轨迹表示为 $T = \left(p_1, p_n\right)$。简化子轨迹的 Balance 值定义如下：

$$Balance\left(T_{sim}\right) = \frac{sed_error_{\varepsilon}}{sed_error_{\max}} + \frac{size_{\varepsilon} - 2}{size_{\max} - 2}$$

$sed_error_{\max}$ 为最简轨迹，即 $T = \left(p_1, p_n\right)$ 到原始轨迹中某一点的最大 SED 距离，$size_{\max}$ 是原始轨迹的大小。sed_error_{ε} 是在简化过程中从未简化的原始轨迹中的某一点到已简化轨迹之间的最大 SED 距离，伴随着轨迹简化的进度将不断变化。$size_{\varepsilon}$ 是在简化过程中随之对应的简化轨迹的大小，同样随着轨迹简化不断变化。ε 的增加会增加 error 的值，但减少 size 的值，反之亦然。因此，最小化 balance 值可以得到在 size 和 error 之间达到最佳平衡的阈值 ε。

本文中计算 SED 误差阈值如算法 2 中所示。该算法的基本原理是找到简化轨迹和原始轨迹之间的最大偏差，即原始轨迹中的点和简化轨迹中的点之间最远 SED 距离，并将此值设置为最大 ε，并将该点添加到简化轨迹。因此，最大偏差缩小，简化轨迹的规模增大。然后，计算该简化轨迹的 Balance 值。当误差可以得到最小 Balance 值时，则存储最小 Balance 值和误差值。整个过程重复，直到简化轨迹 T_{sim} 包含原始轨迹 T 中的所有点。最后计算出导致最小 Balance 值的误差值 ε。算法 2 的时间复杂度为 $O\left(\log n\right)$。

算法 2　计算SED误差阈值

输入：原始轨迹 T

输出：合适的误差阈值 ε

$T'=(p_1,p_n)$;

while $|T'|\neq|T|$ **do**

　找到从 T 到 T' 之间SED距离最远的点 p_i;

　将 p_i 添加至 T';

　$balance=Balance(T')$;

　if $balance<balance_{\min}$ **then**

　　$balance_{\min}=balance$;

　　$\varepsilon=error(T')$;

　end

end

return ε;

2.5 导出简化轨迹

本节介绍在经过以上步骤，得出速度-误差表后，使用速度-误差表简化轨迹的方法。在通过速度值将轨迹划分为子轨迹之后，使用 TD-TR 算法[3]进行简化轨迹。TD-TR 算法是对 Douglas-Peucker 算法的改进，Douglas-Peucker 算法使用垂直欧氏距离(PED)计算点 p_i 与近似轨迹之间的误差距离，具有忽略时态数据的局限性，TD-TR 算法通过使用同步欧氏距离(SED)克服了这个限制。从速度-误差表可以得到速度间隔和相应的 SED 误差阈值，根据每个子轨迹的速度间隔，可以从速度-误差表中获得相应的 SED 误差阈值，该误差阈值是用于简化该子轨迹的 TD-TR 算法中使用的参数。最后将简化好的子轨迹按照时间顺序全部连接，得到最终的简化轨迹 T_{sim}。

TD-TR 算法的时间复杂度为 $O(n^2)$，所以文中的 STS 算法的时间复杂度为 $O(n^2)$。

3 实验与结果

3.1 实验配置及数据集

在本节中，进行了大量的实验。具体配置如下：处理器为 Intel(R) Core™ i5-6300HQ，CPU@2.30 GHz 8 GB RAM 中实现；实验环境为 Windows 10 操作系统和 Anaconda 3 开发环境，Python 3.6 实现。实验数据从项目 Microsoft GeoLife 得到[17, 18]，Microsoft GeoLife 的位置数据是在两年多时间中（2007 年 4 月到 2009 年 8 月）从 178 个人中获得的。数据集中包括各种运输方式，如骑自行车，步行和铁路。大约 91%的轨迹采样率为 1-5 秒，大部分数据的收集发生在中国北京附近，少量在美国和欧洲。

表 3 数据集

序号	名称	轨迹数量（条）	轨迹点数目（个）
1	微小数据集	865	10-300
2	小数据集	660	300-800
3	中数据集	515	800-1600
4	大数据集	822	1600-6400
5	超大数据集	201	＞6400

为了确定算法对于简化性能与轨迹长度之间的关系，将原始轨迹分为 5 类：微短轨迹，短轨迹，中等轨迹，长轨迹和超长轨迹。微短轨迹中每个轨迹有 10-300 个点。短轨迹中每个轨迹有 300~800 个点。中等轨迹中每个轨迹有 800~1600 个点。长轨迹中每个轨迹有 1600~6400 个点。超长轨迹中每个轨迹有 6400 多个点。通过这五类轨迹，如表 3 所示，将原始数据集也分为五类：微小数据集，小数据集，中数据集，大数据集和超大数据集。序列号为 1~5，对不同规模数据集下的压缩时间、简化率、SED 误差、速度误差和空间误差进行测试。

首先对本文所提出的 STS 算法与 ATS 算法在压缩时间和简化率等方面进行了比较，说明了 STS 算法能够在简化率持平的情况下减少压缩时间。然后，对于 STS 算法、ATS 算法、Douglas-Peucker 算法（简称 DP 算法）及 TD-TR 算法，比较了四个算法之间的平均 SED 误差、速度误差以及空间误差。

3.2 度量指标及算法比较

3.2.1 基于简化时间及简化率的评估

简化时间是指原始轨迹段进行简化过程的运行时间，是衡量轨迹简化效率的重要依据，在实际应用中，简化时间越短越好。图 4 描述了 STS 算法与 ATS 算法的实际执行时间，STS 算法的平均压缩时间比 ATS 算法短，并且伴随着数据集的增大，差距愈加明显。

简化率（R）是反应轨迹数据大小变化的性能指标，其定义为 $R=1-\dfrac{n}{m}$，其中 m 代表原始轨迹的轨迹点数目，n 代表简化后轨迹的轨迹点数目。图 5 对两种算法的简化率进行了对比，随着原始轨迹规模的增大，ATS 与 STS 算法的平均简化率均会提高，两种算法的简化率总体相差不大。当原始轨迹较小时（800 个点以下），STS 的简化率比 ATS 略高，但当轨迹较大时，STS 算法的简化率略低于 ATS 算法。

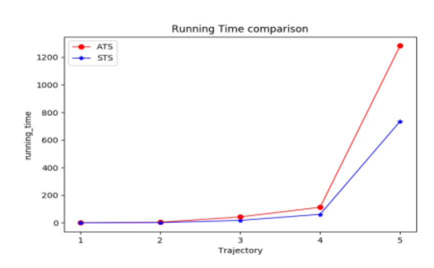


图 4 不同规模数据集下两种算法平均压缩时间的比较

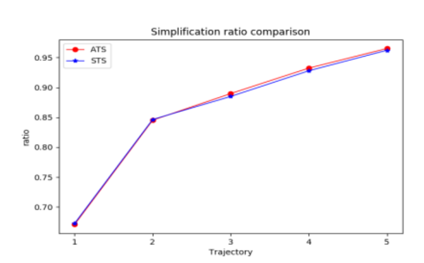


图 5 不同规模数据集下两种算法平均简化率的比较

3.2.2 基于误差度量的评估

SED 误差是指原始轨迹点 p_i 与简化后轨迹中的估计点 $p_i{'}$之间的距离（即 SED 距离）， $p_i{'}$坐标位置的计算方式与 2.4 节中相同。如图 6 所示，原始轨迹 $T=(p_1,p_2,\ldots,p_8)$ 及简化轨迹 $T_{sim}=(p_1,p_4,p_6,p_8)$， p_2 的估计点是 $p_2{'}$，则 p_2 的 SED 误差是 p_2 与 $p_2{'}$的距离。

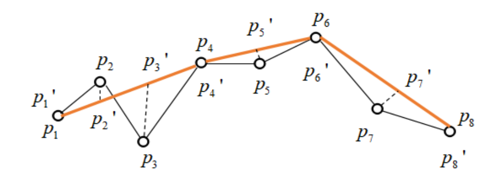


图 6 SED 误差示例

空间误差与 SED 误差类似，只是 SED 误差中计算 p_i 与 $p_i{'}$之间的 SED 距离，空间误差则计算其垂直欧氏距离。

给定原始轨迹点 $p_i=(x_i,y_i,t_i)$ 和 $p_{i+1}=(x_{i+1},y_{i+1},t_{i+1})$，

p_i 的速度计算为 $\dfrac{d(p_i,p_{i+1})}{t_{i+1}-t_i}$。对于 p_i 和 p_{i+1} 的估计点 $p_i{'}$和

$p_{i+1}{'}$，类似地计算 $p_i{'}$处的速度。相对于 p_i 的速度误差被定义

为原始轨迹点 p_i 的速度与简化后轨迹点 $p_i{'}$的速度差值。

SED 误差、速度误差及空间误差是衡量轨迹简化算法性能的重要依据。图 7~9 分别描述了 STS、ATS、DP 和 TD-TR 四种算法在 SED 误差、速度误差、空间误差方面的比较。由图 7 可得，随着数据集的增大，四种算法的 SED 误差均会增大，本文所提出的 STS 算法的 SED 误差增幅远远低于 ATS 和 DP 算法，当数据集较小时 STS 的 SED 误差最小，但在数据集较大时 TD-TR 算法的 SED 误差最小。

图 8 中四种算法的速度误差均会随着轨迹规模的增大而增大，在数据集较大时，DP 与 ATS 算法的速度误差增幅较大，STS 与 TD-TR 算法的速度误差大致相同，在大数据集中 TD-TR 算法速度误差更小，但在小数据集中 STS 优于 TD-TR 算法。

由图 9 可以得到以下结论：STS 算法的空间误差在小数据集中最小，但在大数据及中略高于 TD-TR 算法。

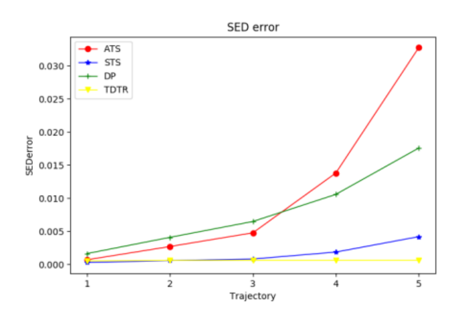


图 7 平均 SED 误差

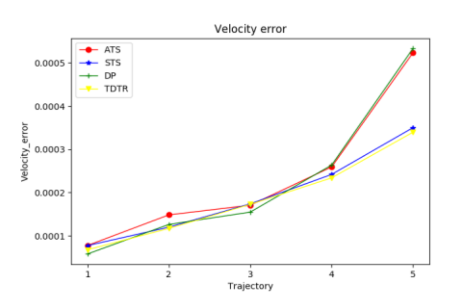


图 8 平均速度误差

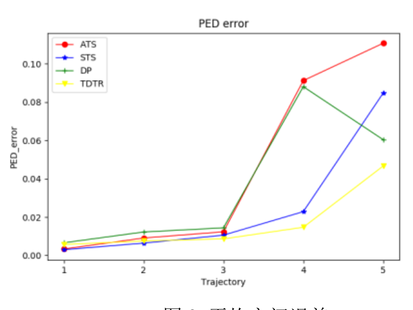


图 9 平均空间误差

4 结束语

本文研究移动对象原始轨迹的简化轨迹的简化问题。在ATS算法的基础上，提出了基于分段的轨迹简化算法STS算法。STS算法首先通过聚类方法从速度值的分布中导出代表性速度区间，通过其代表性速度值，将整个轨迹划分成若干子轨迹。然后分别计算各个子轨迹段的SED误差阈值。最后使用TD-TR算法导出简化轨迹。为了验证本文提出的算法的有效性，使用真实轨迹数据集进行大量的实验。实验表明，STS算法在压缩时间、简化率、平均SED误差、平均空间误差等方面优于ATS算法，总体来说，STS在微小数据集中的表现更好。在接下来的研究中，为了满足基于位置服务工具对移动对象数据分析处理的需求，将进一步与深度学习相结合，考虑如何对移动对象轨迹数据进行预测等一系列处理，研究预测模型，将简化前后轨迹放入预测模型做对比衡量。

参考文献：

[1] Canalys. Worldwide mobile navigation device market more than doubles [R]. [S. l.] : Canalys Research Release, 2007.

[2] Canalys. North America overtakes EMEA as largest satellite navigation market [R]. [S. l.] : Canalys Research Release, 2009.

[3] Wilson B D, Mannucci A J, Edwards C D. Subdaily northern hemisphere ionospheric maps using an extensive network of GPS receivers [J]. Radio Science, 2016, 30 (3): 639-648.

[4] Lichman M, Smyth P. Modeling human location data with mixtures of kernel densities [C]// Proc of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2014: 35-44.

[5] Song Renchu, Sun Weiwei, Zheng Baihua, *et al.* PRESS: a novel framework of trajectory compression in road networks [J]. Proceedings of the VLDB Endowment, 2014, 7 (9): 661-672.

[6] Sun Penghui, Xia Shixiong, Yuan Guan, *et al.* An overview of moving object trajectory compression algorithms [J]. Mathematical Problems in Engineering, 2016, 2016 (3): 1-13.

[7] Muckell J, Olsen P W, Hwang J H, *et al.* Compression of trajectory data: a comprehensive evaluation and new approach [J]. Geoinformatica, 2014, 18 (3): 435-460.

[8] He Xinran, David Kempe. Stability of influence maximization [J]. 2015: 1256-1265.

[9] Douglas D H, Peucker T K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature [J]. Cartographica the International Journal for Geographic Information & Geovisualization, 1973, 10 (2): 112-122.

[10] Meratnia N, By R A D. Spatiotemporal Compression Techniques for Moving Point Objects [C]// Proc of International Conference on Extending Database Technology. 2004: 765-782.

[11] Cheng Long, Wong R C, Jagadish H V. Direction-preserving trajectory simplification [J]. Proceedings of the VLDB Endowment, 2013, 6 (10): 949-960.

[12] Cheng Long, Wong R C, Jagadish H V. Trajectory simplification: on minimizing the directionbased error [J]. Proceedings of the VLDB Endowment, 2014, 8 (1): 49-60.

[13] Deng Ze, Han Wei, Wang Lizhe, *et al.* An efficient online direction-preserving compression approach for trajectory streaming data [J]. Future Generation Computer Systems, 2017, 68: 150-162.

[14] Lin Chihyu, Hung C C, Lei P R. A velocity-preserving trajectory simplification approach [C]// Technologies and Applications of Artificial Intelligence. 2017: 58-65.

[15] 朱连江，马炳先，赵学泉. 基于轮廓系数的聚类有效性分析 [J]. 计算机应用，2010, 30 (s2): 139-141. (Zhu LianJiang, Ma Bingxian, Zhao Xuequan. Clustering validity analysis based on silhouette coefficient [J]. Journal of Computer Applications, 2010, 30 (s2): 139-141.)

[16] 陈辉林，夏道勋. 基于 CART 决策树数据挖掘算法的应用研究 [J]. 煤炭技术，2011, 30 (10): 164-166. (Chen Huilin, Xia Daoxun. Applied research on data mining based on CART decision tree algorithm [J]. Coal Technology, 2011, 30 (10): 164-166.)

[17] Zheng Yu, Li Quannan, Chen Yukun, *et al.* Understanding mobility based on GPS data [C]// Proc of International Conference on Ubiquitous Computing. New York: ACM Press, 2008: 312-321.

[18] Zheng Yu, Zhang Lizhu, Xie Xing, *et al.* Mining interesting locations and travel sequences from GPS trajectories [C]// Proc of International Conference on World Wide Web. New York: ACM Press, 2009: 791-800.